A Hybrid Bi-LSTM-CRF Model for Sequence Labeling Applied to the Sourcing Domain

Hasnaa Daoud[1], Molka Tounsi Dhouib[1,2], Jerôme Rancati[1],
Catherine Faron[2], Andrea G. B. Tettamanzi[2]

[1] Silex France
[2] Université Côte d'Azur, Inria, CNRS, I3S, Sophia Antipolis, France

{hasnaa.daoud, molka.tounsi, jerome.rancati}@silex-france.com,
{dhouib,faron,tettaman}@i3s.unice.fr

Résumé

Dans un certain nombre de domaines, les entreprises sont souvent confrontées à la tâche de traiter au quotidien des quantités importantes de demandes textuelles. L'extraction automatique des informations clés à partir des demandes clients, peut aider à accélérer le processus de traitement. Silex France est aujourd'hui confrontée à ces enjeux dans le cadre du traitement des demandes de sourcings.
Dans cet article, nous partageons nos résultats d'étiquetage de séquences en nous basant sur une méthode hybride BiLSTM-CRF, dans un contexte industriel. Le travail est intégré dans la plate-forme B2B Silex pour la recommandation des prestataires de services. Les expériences faites sur les données de la plateforme B2B Silex montrent qu'avec un bon choix de features à extraire et des hyperparamètres, la combinaison du modèle Bi-LSTM-CRF permet de réussir l'extraction d'infomation à partir des demandes textuelles, même dans un contexte de petites données (small data). En effet, le contenu textuel traité est sous forme de phrases complètes générées par des utilisateurs, et est ainsi exposé à des erreurs de frappe. Pour gérer ce type de données, nous combinons plusieurs types de features extraites décrivant le contenu textuel tels que : (i) la sémantique, (ii) la syntaxe, (iii) les caractères des mots, (iv) la position des mots.

Mots-clés

Traitement du Langage Naturel, Extraction d'Information, Etiquetage de Séquences, Réseaux de neurones artificiels

Abstract

In a number of areas, companies are often faced with the task of dealing with large amounts of textual customers' requests. Automating information extraction like key phrases from customers' requests can help to accelerate the processing process. Silex France is currently facing this challenge in the context of processing sourcing requests.
In this article, we share our sequence labeling results based on a hybrid method Bi-LSTM-CRF, in an industrial context. This work was integrated in the B2B Silex platform for service providers recommendation. Experiments with the B2B Silex platform data show that, with a good choice of features to extract and optimal choice of hyperparameters, the combination of the Bi-LSTM and CRF helps to achieve good results even in a context of small data. Indeed, the textual content processed is in the form of complete sentences generated by users, and thus is subject to typing errors. To handle this type of data we combine several types of extracted features describing the textual content such as : (i) semantics, (ii) syntax, (iii) word characters, (iv) position of words.

Keywords

Natural Language Processing, Information Extraction, Sequence Labeling, Artificial Neural Networks

1 Introduction

Natural Language Processing (NLP) is a branch of Artificial Intelligence (AI) that helps computers understand, interpret and manipulate human language. Information Extraction (IE) is a crucial task in the field of NLP and linguistics that is widely used for tasks such as Question Answering, Machine Translation, Entities Extraction, Event Extraction...

In this paper, we report on an IE task we conducted in the context of *Silex* company which develops a SaaS e-sourcing platform for the identification of the best providers that meet expressed needs. It takes into consideration the requested service, costs, deadlines, innovation, and quality [4]. There are two main user communities within *Silex* platform : (i) service providers, and (ii) service buyers. Silex aims to automatically analyze the textual descriptions of service requests and providers in order to better evaluate opportunities in a faster way with more targeted sourcings.

We aim to extract key phrases from customers' requests. In the context of sourcing requests, a key phrases usually indicates a product, a service, an occupation (job title) or a skill. These are the main entities in Sourcing domain. These types of information can be considered as *contextualized* named entities. In fact, it turns out that in some requests, a customer may talk about his own services, but

we aim to extract only the services he needs.

We propose an IE approach based on a Bi-LSTM-CRF architecture, able to analyze textual descriptions (service providers and service requests) and extract the relevant parts of the text that summarize a provider's offer/ a customer need (such as services, products, occupations, skills). In this paper, we focus on information extraction from service requests.

Processing of service requests is more challenging than that of service offers : (i) text requests are generally short (50 words on average in our case) ; (ii) the content of these descriptions is user-generated, and then is subject to typing errors ; (iii) finally, a user may describe his own products or services to contextualize his request, which would create confusion. This raises the issue of distinguishing between the real user's need and the general context of the sourcing request description. Therefore the task is not the extraction of *any* expression describing a service or a product in a sourcing request.

Our main research question is : How to extract relevant information that summarizes a customer's need ? We focus on the following sub-questions :

- Which is the best approach to extract information from short texts ?
- Which types of embeddings must we use to extract relevant information in our case ?
- How to deal with the limited number of data ?

Our approach is based on the Bi-LSTM-CRF framework [11] ; a Bidirectional Long-term and Short-term Mermory (Bi-LSTM) encodes an input sequence words and a Conditional Random Fields model (CRF) labels every sequence word. In [11], the authors use two types of embeddings : *word embeddings* and *character based embeddings* (computed using a Bi-LTSM). The final embeddings, which constitutes inputs of the main Bi-LSTM-CRF are obtained by concatenating *word embeddings* and *character based embeddings*.

Our contribution lies in the addition of two other types of embeddings computed with two Bi-LSTMs : (i) Syntactic embeddings and (ii) Position embeddings. Also, to answer *Silex* use case, since we have only 858 annotated service requests, we adapted the architecture hyper-parameters to our context of small data.

This paper is organized as follows : Section 2 presents the related works. Section 3 describes our information extraction approach. Section 4 describes our data and our implementation. Section 5 reports and discusses the results of our experiments. Section 6 concludes with an outline of future work.

2 Related Works

There are three common techniques in the literature for the NER task [17, 12] : (i) knowledge-based unsupervised systems, (ii) feature-engineered supervised systems and (iii) feature-inferring neural network systems.

Our work is mostly related to feature-inferring neural network systems. [3] proposes semi-supervised method where a deep neural network model learns internal representations on the basis of vast amounts of mostly unlabeled training data. This model presents two main limitations : (i) it doesn't take into account long-term dependencies between words because it is based on a simple feed-forward neural network and limits the use of context to a fixed-size window ; (ii) by using only word embedding, the model is unable to exploit other features such as letter's cases or complex aspects of user-generated content. These types of considerations could be useful, especially for rare words.

To overcome some of the limitations of [3]'s model, deep learning algorithms such as Recurrent Neural Networks (RNN) are successfully applied for sequence labeling task. Authors of [18] present a detailed comparison between the Convolution Neural Networks (CNN), and RNN in the particular context of NER. The specificity of RNN is that it allows a neural network to exhibit temporal dynamic behavior to process input sequences with no limitation on the input size. However, RNN are not adapted when input sequences are getting too long ; in a RNN, gradients are back-propagated through all time steps as well. This means that the longer our sequence size is, the more gradients we will be taking the product of. This leads to the vanishing gradient problem. Long-term and Short-term Memory Networks (LSTMs) are a particular type of RNN that are designed to avoid this problem through the use of LSTM cells, which makes it easy to learn about long term dependencies [5].

[11] proposes a more powerful neural network model that incorporates a bi-directional LSTM (Bi-LSTM) and CRF. This model is based on robust learning with the dropout, which allows good recognition results of NER. The bi-directional LSTM model takes into account the whole context enables to effectively train a model with the flexible use of long range context [6].

In addition to the architecture, the way we present input data to a neural network matters. For example, we can see a textual sentence in different ways : (i) sequence of words, (ii) sequence of letters, (iii) sequence of chunks... Enriching the input sequences with accessible additional data can also help to have better results. Character-based representations are very important in our use case. In fact, our data are user-generated. Then, it is important to capture morphological and orthographic patterns. [2] presents a hybrid bidirectional LSTM and bidirectional CNN neural network architecture that helps to exploit explicit character level features such as prefixes and suffixes, which could be useful especially with rare words for which word embeddings are poorly (or are not) trained. [14] introduces the neural character embedding in the NER task for English and achieves the state-of-the-art. [1] explored ways to improve point-of-sale labeling using different types of auxiliary losses, and different representations of words. They built their model based on Bi-LSTM layers and showed that introducing word representations through their characters gives better results in terms of model speed and performance. [9] proposes a simple yet effective dependency-guided LSTM-CRF model that takes the complete depen-

dency trees and captures syntactic properties for the Named Entities Recognition task. Furthermore, [11] incorporated character-level structure into word representation. Each input vector consists of two parts : (i) pre-trained word-level representation [13] and (ii) task-related character-level representation. The authors of [11] adopted a bidirectional LSTM to capture information in both forward and backward directions and concatenate the outputs of these two LSTMs.

Most related works cited in this paper use only word embedding, or combine them with character based representations [9, 1], while we enrich input sequences with Part Of Speech tagging and word position information. The way we represent sequences improves the results in our use case.

3 Methodology

Bi-LSTM-CRF model is introduced by Huang and al [8]. It has been compared to LSTM, Bi-LSTM and LSTM-CRF models. Best results in the paper are achieved with Bi-LTSM-CRF model in different sequence tagging tasks (Part-of-Speech Tagging, Chunking, and NER tasks). In the first part of this section, we present the state of the art Bi-LSTM-CRF model and how we use it in our architecture. In the second part, we detail the features we introduce to enrich word embeddings.

3.1 Architecture

3.1.1 Bi-LSTM :

Bidirectional LSTM or Bi-LSTM model [7] is composed of two LSTMs each one processes the sequence in a different direction.

The main characteristics of LSTM are (i) the ability to operate on sequential data and (ii) the ability to capture long term dependencies thanks to a memory-cell.

LSTM takes as input a sequence of vectors $X = (x_1, x_2, ..., x_n)$ and returns another sequence of vectors, named hidden states vectors $H = (h_1, h_2, ..., h_n)$ as output.

Below, we detail mathematical equations in LSTM network we used in this work :

$$
\begin{aligned}
i_t &= \sigma(W_{xi}x_t + b_{xi} + W_{hi}h_{(t-1)} + b_{hi}) \\
f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{(t-1)} + b_{hf}) \\
C_t &= \tanh(W_{iC}x_t + b_{iC} + W_{hC}h_{(t-1)} + b_{hC}) \\
o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{(t-1)} + b_{ho}) \\
c_t &= f_t \mathbf{x} c_{(t-1)} + i_t \mathbf{x} C_t \text{ [1]} \\
h_t &= o_t \mathbf{x} \tanh(c_t)
\end{aligned}
$$

Where x_t is the input at time t, h_t is the hidden state at time t, c_t is the cell state at time t, h_t is the hidden state of the layer at time t-1 or the initial hidden state at time 0, and i_t, f_t, C_t , o_t are the input, forget, cell, and output gates, respectively. σ is the sigmoid function.

Figure 1 presents the LSTM architecture, where pink circles represent arithmetic operators and rectangles represent LSTM gates.

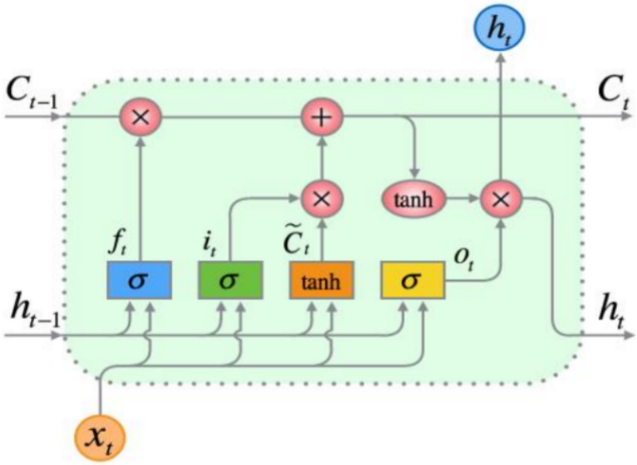


FIGURE 1 – LSTM Architecture.

The drawback of the LSTM model is that it processes input from left to right, which involves that it can only encode dependencies on previous tokens. That is why a second LSTM is used to process input in the reverse direction (i.e., from right to left). This new layer makes it possible to detect dependencies on the right context of a token.

In our model, we use Bi-LTSMs to :

— Extract character based embeddings, syntactic representations, and position representations. This part will be detailed in section 3.2.

— Combine all of these representations to extract complete dependency information. The idea is to concatenate the above representations and pre-trained word embeddings to have complete representations of words, then to pass the obtained representations by a Bi-LSTM.

3.1.2 Linear Conditional Random Field

The way sequences are encoded is not the only important part in a sequence labeling problem. The chosen classifier also plays a crucial role. One simple approach is to classify each word independently. The problem with this approach is that it assumes that given the input, all of the entity labels are independent.

In order to relax this assumption, *Generative Models* named HMMs (Hidden Markov Models) make two assumptions : (i) Each state depends only on its immediate predecessor. (ii) Each observation variable depends only on the current state. This last statement makes it impossible to add additional knowledge in the model (e.g contextual information). Using conditional Random Fields (CRF) [10] is a solution to overcome this issue. CRFs are undirected graphical models and are partially similar to HMMs. Nonetheless, they are not *Generative* but *Discriminative Models* trained to maximize the conditional probability of observation and state sequences [15]. The primary advantage of CRFs over HMMs is their conditional nature, resulting in the relaxation of the independence assumptions required by

1. **x** is Hadamard product

HMMs in order to ensure tractable inference.
In our Bi-LSTM-CRF model, after extracting a sequence hidden vectors $(h_1, h_2, h_3, ..., h_n)$ with Bi-LSTMs (with n the sequence length), we compute scores associated with each label j at position t in the sequence with a linear layer. We consider P the resulted matrix of size $n \times k$ (with k the number of labels) :

$$P_t = W_{hp} h_t + b_{hp} \forall t \in [[1, n]] \quad (1)$$

with :

— $h_t = (h_{t_l}, h_{t_r})$ hidden vector at position t in the sequence (concatenation of two hidden vectors ; right context hidden vector and left context hidden vector)
— W_{hp} weight matrix of dimension (k, m) (with m the hidden vectors size)
— b_{hp} bias vector of dimension k

As explained earlier, CRF model does not rely only on matrix P since we assume that predicted variables in a sequence depend on each other.
Inference in linear CRF models is based on maximizing the following conditional probability :

$$P(y|H) = \frac{\exp(score(H, y))}{\sum_{y\prime \in Y} \exp(score(H, y\prime))} \quad (2)$$

with :

— H the matrix of hidden vectors : $H = [h_1, h_2, h_3, ..., h_n]$
— $P(y|H)$ the conditional probability of a sequence of tags y .
— Y the set of possible tag sequences.

$score$ function is defined as follows :

$$score(H, y) = \sum_{t=1}^{n} A_{y_t, y_{t+1}} + \sum_{t=1}^{n} P_{t, y_t} \quad (3)$$

With A the transition scores matrix ($A_{i,j}$ transition score from state i to state j)

3.2 Features

Our main contribution in this paper is the addition of new kinds of features extracted using Bi-LSTMs. Our model uses four different kinds of embeddings for each sequence word. Three types of these embeddings are trained with Bi-LSTMs, they are concatenated with pretrained *FastText* word embeddings, and used as the input sequence to our main Bi-LSTM-CRF model.
Figure 2 schematizes how the different types of embeddings are trained.

3.2.1 Word embedding

Word embeddings are vector representations trained on a large corpus of texts such as encyclopedias, news texts, or literature. There are many language modeling and feature learning techniques that help to map words to vectors that allow to represent the contextual proximity of different words. It is possible to train from scratch these vector representations on the particular task of keyword sequence tagging using our data. However, it turns out that the size of our data will not allow us to cover a large vocabulary. Therefore, we chose to use pre-trained word vectors for French, learned using fastText [2] on a Wikipedia dump. The French model we used contains 1.152.449 words mapped to 300-dimensional vectors [13].

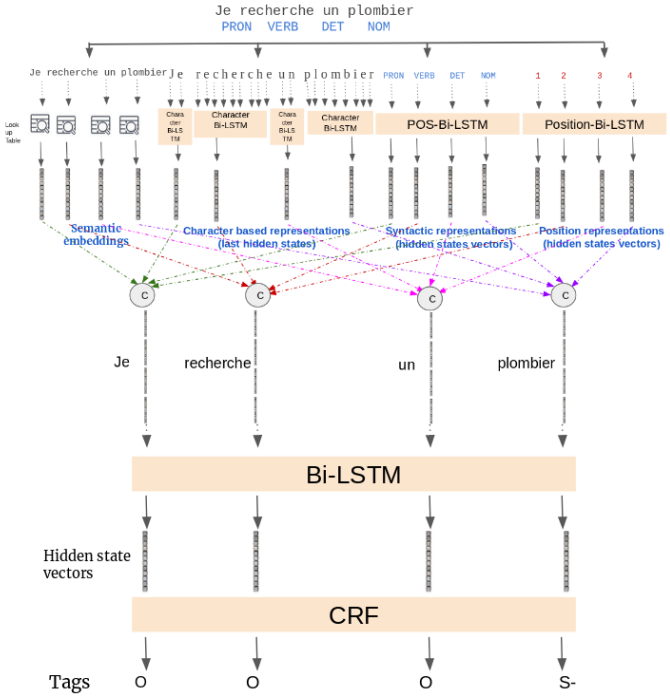


FIGURE 2 – Model architecture (Embeddings Extraction + Main BiLSTM-CRF)

3.2.2 Syntactic Word Representations

Manual analysis of our data shows that syntax plays an important role to locate the user's need in a sequence. Hence the need to extract features representing syntax and dependency between words. This type of information (i.e., syntactic structure and dependencies) is important to complete the semantic information. For example, many sourcings requests contain verbs like (*rechercher, souhaiter, chercher ...*), in these cases recognizing the sentence object would help the model to recognize the customer's need. We then trained part-of-speech (POS) embeddings, using a Bi-LSTM Model.

3.2.3 Character-level Word Representations

Fasttext pretrained Word embedding allows to extract information about the meaning of words, but the covered vocabulary remains limited to the vocabulary of the training corpus. Therefore, rare words or words with spelling errors cannot be represented. Hence the importance of Character-level Word Representations since any word is made up of a number of characters and characters set is finite. For every word, we use a BiLSTM that takes as input the sequence of word's characters, and returns the last hidden states vector. We consider this vector as a character level based represen-

2. https ://fasttext.cc/docs/en/pretrained-vectors.html

tation.

3.2.4 Position Representation

This kind of information is important in our use case : based on manual analysis of our data, it turns out that the main subject is mentioned at the beginning of the text. We are convinced that this is generally valid for written text requests. Thus, it would be interesting if the model takes into account the position of words. This way, the model will be able to understand that it is highly likely that the words at the beginning of the text are relevant information. We use a Bi-LSTM model to extract this type of embedding.
All these types of representations are concatenated and used as input of the main Bi-LSTM-CRF bloc as shown in Figure 2.

4 Experiments

4.1 Dataset

Our dataset is composed of 883 descriptions of service requests distributed as follows : (i) 594 service request descriptions in the train set, (ii) 198 service request descriptions in the development set, and (iii) 90 service request descriptions in the test set.
Data are annotated by Sourcing experts at *Silex* according to the BIOES format, which stands for Beginning, Inside, Outside, End, and Single to indicate the position of a token in a tagged segment. Descriptions are annotated as follow :

- **B-** : to mark the beginning of an entity
- **I-** : to mark the inside of an entity
- **E-** : to mark the end of an entity
- **S-** : to mark a single entity
- **O** : to mark a token outside all of entities

Table 1 shows an example annotation using the BIOES format.

Word	POS	Label
Je	PRON	O
recherche	VERB	O
un	DET	O
plombier	NOUN	S-
La	DET	O
société	NOUN	O
BNB	NOUN	O
souhaite	VERB	O
créer	VERB	O
des	DET	O
supports	NOUN	B-
de	ADP	I-
communication	NOUN	E-
.	.	.
.	.	.

TABLE 1 – Example of input training data

Artificial neural networks generally require a large corpus of training data in order to learn efficiently the task. To overcome this limitation of the training set size (594 service requests), we augmented the training set. We generated new training examples by applying some transformations of the available ones. These transformations must preserve the entities' labels.
In the field of NLP, it is difficult to increase text due to the complexity of natural language. Some methods in the literature use shuffling, which involves changing the order of sentences in the text or random deletion of certain words from the text. However, this augmentation techniques may change the whole context of the text. In natural language processing, it is not a good idea to augment data using signal transformation techniques (images [16], speech recognition...), because the order of characters is important to keep a strict syntactic and semantic meaning. Other augmentation techniques make more sense in the context of our work, such as injecting punctuation noise or modifying certain characters in words. Otherwise, it is difficult to add more semantics, the best way to do so is to use human sentence rephrasing, but it is an expensive operation. Therefore, the most natural choice in data augmentation for us is to replace words with their synonyms based on a dictionary of synonyms of the most frequent words in the field of sourcing.
We therefore chose to augment our training dataset by :

- introducing punctuation noise
- changing characters of some words (which simulates spelling errors in user-generated data)
- replacing some words with their synonyms

This way, we multiplied our training set size by three.

4.2 Implementation

To implement our model, we used the Pytorch library, which supports GPU computing. For all Bi-LSTMs in our model, we used the *torch.nn.LSTM* class, which allows to create LSTMs and to set parameters such as the number of layers, bidirectionality, input feature size, hidden state vector size. We implemented the CRF based on equations of section 3.1.2. We use the *Viterbi* algorithm for finding the most likely sequence of hidden states.

4.3 Hyper-parameters

Table 2 shows the hyper-parameters we use in Bi-LSTM layers used for character-based, position and syntactic features extraction.

Features	**Embedding dimensions**	**Hidden vectors dimensions**
Syntactic features	25	30
Character-level based features	25	50
Position features	25	30

TABLE 2 – Hyper-parameters for the features extraction layers

For the final Bi-LSTM bloc (see Figure 2) used to combine all the features of Table 2 with Fasttext word embeddings, we chose a hidden layer of size 400.
In our experiments, we started by initializing the semantic word embedding layer with FastText pre-trained embeddings, and we updated them during training. We found that

this causes overfitting because of a high number of parameters. We then chose to **freeze the semantic word embedding layer**, which helped us to improve the results.
In addition to increasing the data, to avoid over-fitting, we chose a high dropout value of 0.5.
In the literature, the number of Bi-LSTMs hidden layers is usually equal to the same number of embedding units. We compared the results with different units' numbers in the main Bi-LSTM and we were able to get better results when the size of the hidden layers is equal to 400.
In this paper, we conducted four experiments to compare the performance of four kinds of models :

— I : Bi-LSTM model with only word embedding and logistic regression classification model
— II : Bi-LSTM model with only word embedding and CRF classification model
— III : Bi-LSTM-CRF model with word embedding, character-based representations, and Bi-LSTM position based representation.
— IV : Bi-LSTM-CRF model with word embedding, character-based representations, Bi-LSTM position-based representation and syntactic word representations.

5 Result and Discussion

The problem we deal with is different from ordinary NER tasks, since we detect text segments summarizing a user's need. Let us note that even expert annotators find it sometimes hard to decide on the segment to annotate. Thus, we chose to evaluate the four models in two different ways : (i) Precision and Recall, (ii) Cosine similarity.

5.1 Precision and Recall

We started with an evaluation based on precision, recall, and the F1 score as the basis for choosing the best model. In our evaluation, we do not consider the complete annotated entity, but rather words of the entity separately. For example, in the sentence *"I am looking for a **plumber** able to **repair a faucet Sprinkle**"*, we do not fully penalize the algorithm if it does not detect the complete **repair a faucet Sprinkle** segment. But we count words that it could detect in that segment. Indeed, if we suppose that the model detects only **repair a faucet**, this may be enough to understand the need's subject. We also ignore conjunctions, determinants and punctuation in the evaluation. Table 3 presents the results obtained in terms of precision, recall and F_1 score.

	Recall		Precision		F1	
Model	Dev	Test	Dev	Test	Dev	Test
I	58.88	61.31	77.02	76.61	66.74	68.11
II	64.03	66.61	75.21	76.66	69.17	71.29
III	63.06	66.61	75.62	**80.11**	68.77	**72.74**
IV	67.57	**70.20**	76.03	73.77	**71.55**	71.94

TABLE 3 – Precision, Recall and F1-score without data augmentation

Figure 3 and Figure 4 respectively show the evolution of the F1 score and the evolution of the loss function across epochs.

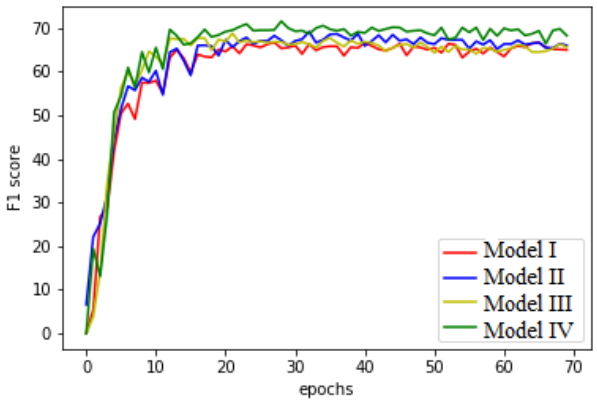


FIGURE 3 – Evolution of score $F1_{Dev}$

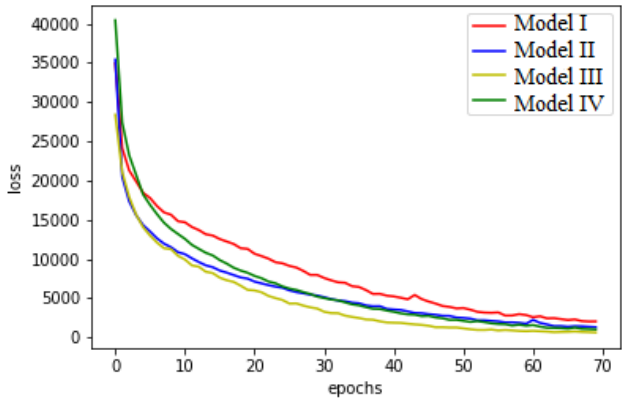


FIGURE 4 – Evolution of the function (*loss*)

One can see that model IV which uses all types of features is the best model across all epochs. Model I, which takes as word representations only word embedding with logistic regression for tagging, has the lowest score. Figure 4 shows that from epoch 30, the loss function continues to decrease without improving the F1 score. From this epoch, the model starts to overfit on training data.
Using syntactic information with POS tagging significantly improves Dev and Test recall, and balances well precision and recall (see Table 3). We also note that with this model, we were able to get a similar F1 scores in dev and test data (difference of 0.39%). The switch from CPU to GPU computing allowed us to gain in terms of performance (more than 1 difference for the $F1\ score$, which is due to the difference between implementations of the libraries used on CPU and GPU) and in terms of training time (from 350 seconds per epoch to 75 seconds per epoch for model IV) .
With data augmentation (Table 4), we have mainly improved the $F1_{Dev}$ scores of the different models. The recall

	Recall		Precision		F1	
Model	Dev	Test	Dev	Test	Dev	Test
I	60.94	61.93	77.31	73.79	68.15	67.35
II	65.64	67.71	75.50	70.23	70.22	68.94
III	63.64	**68.49**	75.28	72.68	70.13	70.52
IV	67.05	68.02	75.34	**77.58**	**70.96**	**72.49**

TABLE 4 – Results with data augmentation

associated with test dataset was relatively improved for the first three models, but the precision decreased relatively, except for model IV.

We can deduce that the best model in terms of Test and Dev scores is model IV that uses a Bi-LSTM-CRF applied on the concatenation of semantic embedding, POS embedding, Character based embedding, and Position embedding.

We point out that the difference between dev and test scores is due to the small amount of annotated data.

The disadvantage of this evaluation method in the particular case of our work is that it gives equal importance to all extracted words. However, it turns out that some words are more important than others, especially words for which meaning appears several times in a service request extracted segments. In the sentence *"I am looking for a **plumber** able to **repair a faucet Sprinkle**"*, "plumber" and "faucet" are two words that generally appear in the same context and could be considered as the most important extracted words. "Sprinkle" ; the faucet brand, is the least important word. However, with precision and recall evaluation, all words are given the same weight, and the model will be penalized in the same way if it does not detect the word **Sprinkle** or the word **plumber**.

We mentioned earlier that the manual annotation was not obvious to the experts. Here is a second possible scenario : in the previous example, an annotator decides to annotate only "plumber" as a unique key phrase, and the model annotates only "repair a faucet", even if there is an important semantic similarity between the two segments, the model will be penalized in terms of Precision and Recall scores. Hence, to deal with this issue, we propose a meaning-based evaluation.

5.2 Evaluation using cosine similarity distribution

To address all of the above problems, we tried to present the results otherwise, which corresponds to the purpose of this work.

Indeed, the goal is to be able to reduce a service request description into a set of keywords. So we thought of calculating the cosine similarity between embedding of expert annotated segments and embedding of segments detected by the algorithm for each service request, and then plot the histogram of the results. A service request extracted segments' embedding is computed by averaging on FastText pretrained word embeddings of their words. We ignore stop words and punctuation. In Figure 5, we present four histograms each associated to a different model.

Note that each type of features added to the model helps to move the distribution a little bit to the right. We can clearly see the difference between the distribution of model I and the distribution of model IV : the distribution of model IV is tightest around 1, with a highest peak.

6 Conclusion

In this paper, we proposed a method that relies on Bi-LSTM-CRF for sequence labeling to summarize sourcing requests. We combined several types of features to represent every word in a sequence. The key of success of our method is an original combination of input features. In addition to word embedding, we extract three other kinds of embeddings : (i) character-level based embeddings, (ii) syntax based embeddings and (iii) position embeddings. These additional embeddings are extracted using Bi-LSTMs and are concatenated with word embedding. We have shown that syntax and position of words help to improve the quality of the information extraction in our use case. We also shared hyper-parameters that give us the best training and choices we made to overcome overfitting problems. Moreover, we have shown that Bi-LSTM-CRF architectures for information extraction can provide value even in a small data context.

This work was integrated into Silex sourcing platform to recommend similar service requests, which considerably reduces the processing time in 60% of cases. This recommendation is based on semantic similarity between requests based on their extracted segments.

As future work we aim to experiment new extraction approaches based on transformers like BERT or Camembert for French texts. We also aim to evaluate the generality of our approach designed for the sourcing domain by experimenting it in a general benchmark.

Références

[1] Daniil Anastasyev, Ilya Gusev, and Eugene Indenbom. Improving part-of-speech tagging via multitask learning and character-level word representations. *arXiv preprint arXiv :1807.00818*, 2018.

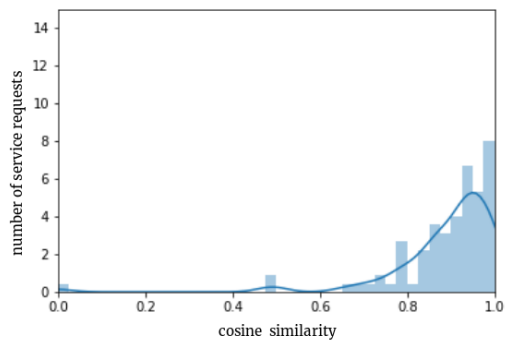
[2] Jason PC Chiu and Eric Nichols. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics*, 4 :357–370, 2016.

[3] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(Aug) :2493–2537, 2011.

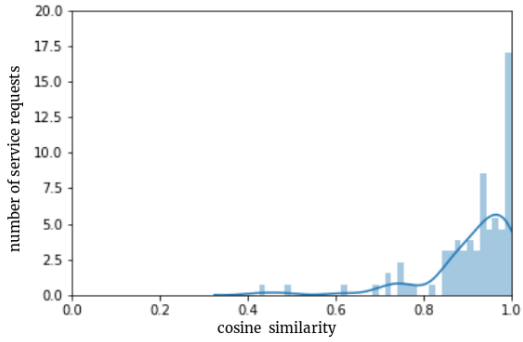
[4] Rémi ESCHENLAUER. Le sourcing, June 2013.

[5] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget : Continual prediction with lstm. 1999.

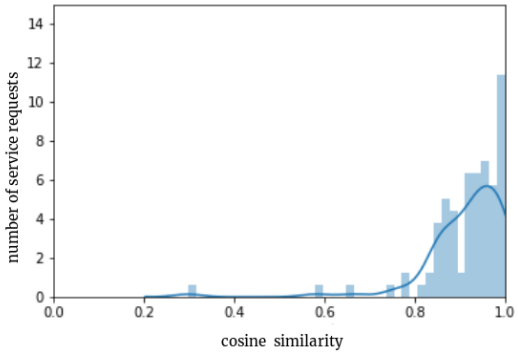
[6] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference*



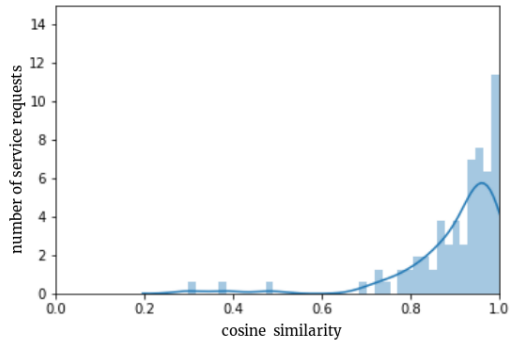
(a) Model I



(b) Model II



(c) Model III



(d) Model IV

FIGURE 5 – Distribution of cosine similarity between segments automatically detected and segments annotated by human experts

on acoustics, speech and signal processing, pages 6645–6649. IEEE, 2013.

[7] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural networks*, 18(5-6) :602–610, 2005.

[8] Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv :1508.01991*, 2015.

[9] Zhanming Jie and Wei Lu. Dependency-guided lstm-crf for named entity recognition. *arXiv preprint arXiv :1909.10148*, 2019.

[10] John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields : Probabilistic models for segmenting and labeling sequence data. 2001.

[11] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. *arXiv preprint arXiv :1603.01360*, 2016.

[12] Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. A survey on deep learning for named entity recognition. *arXiv preprint arXiv :1812.09449*, 2018.

[13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[14] Cicero Nogueira dos Santos and Victor Guimaraes. Boosting named entity recognition with neural character embeddings. *arXiv preprint arXiv :1505.05008*, 2015.

[15] Charles Sutton, Andrew McCallum, et al. An introduction to conditional random fields. *Foundations and Trends® in Machine Learning*, 4(4) :267–373, 2012.

[16] Luke Taylor and Geoff Nitschke. Improving deep learning using generic data augmentation. *arXiv preprint arXiv :1708.06020*, 2017.

[17] Vikas Yadav and Steven Bethard. A survey on recent advances in named entity recognition from deep learning models. *arXiv preprint arXiv :1910.11470*, 2019.

[18] Zenan Zhai, Dat Quoc Nguyen, and Karin Verspoor. Comparing cnn and lstm character-level embeddings in bilstm-crf models for chemical and disease named entity recognition. *arXiv preprint arXiv :1808.08450*, 2018.